Systems Programming

The Memory Hierarchy

Textbook coverage:
Ch 6: The Memory Hierarchy

Byoungyoung Lee
Seoul National University
byoungyoung@snu.ac.kr
https://lifeasageek.github.io

Lecture slides are prepared based on materials provided by CSAPP authors.

Today

- **The memory abstraction**
- Locality of reference
- The memory hierarchy
- Storage technologies and trends

Recall: Writing & Reading Memory

- **Write / Store**
  - Transfer data from memory to CPU

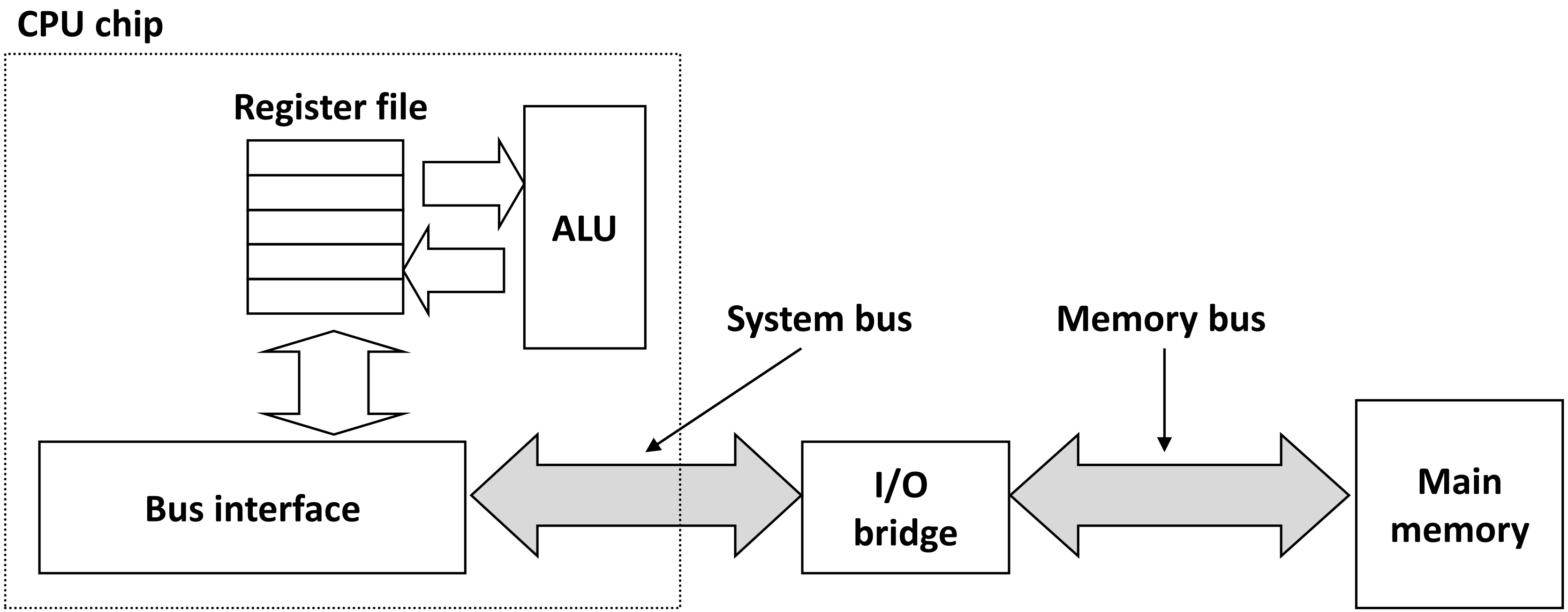
    ```
    movq %rax, 8(%rsp)
    ```

- **Read / Load**
  - Transfer data from CPU to memory

    ```
    movq 8(%rsp), %rax
    ```

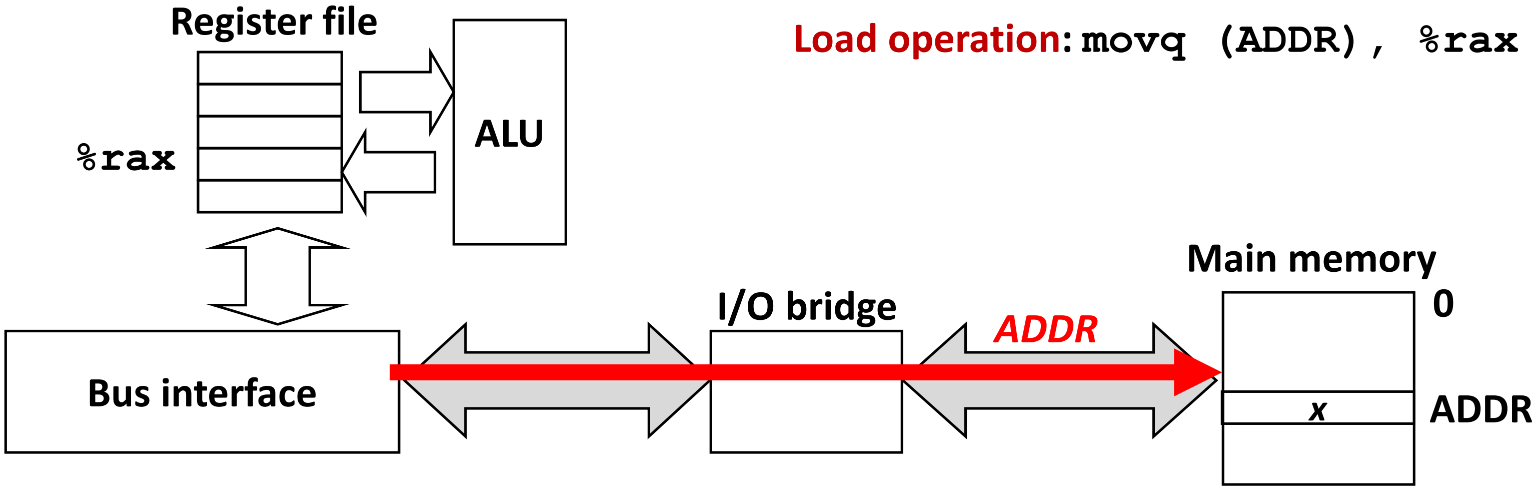
Traditional Bus Structure Connecting CPU and Memory

- A **bus** is a collection of parallel wires that carry followings
  - Address
  - Data
  - Control signals
- **Buses are typically shared by multiple devices.**



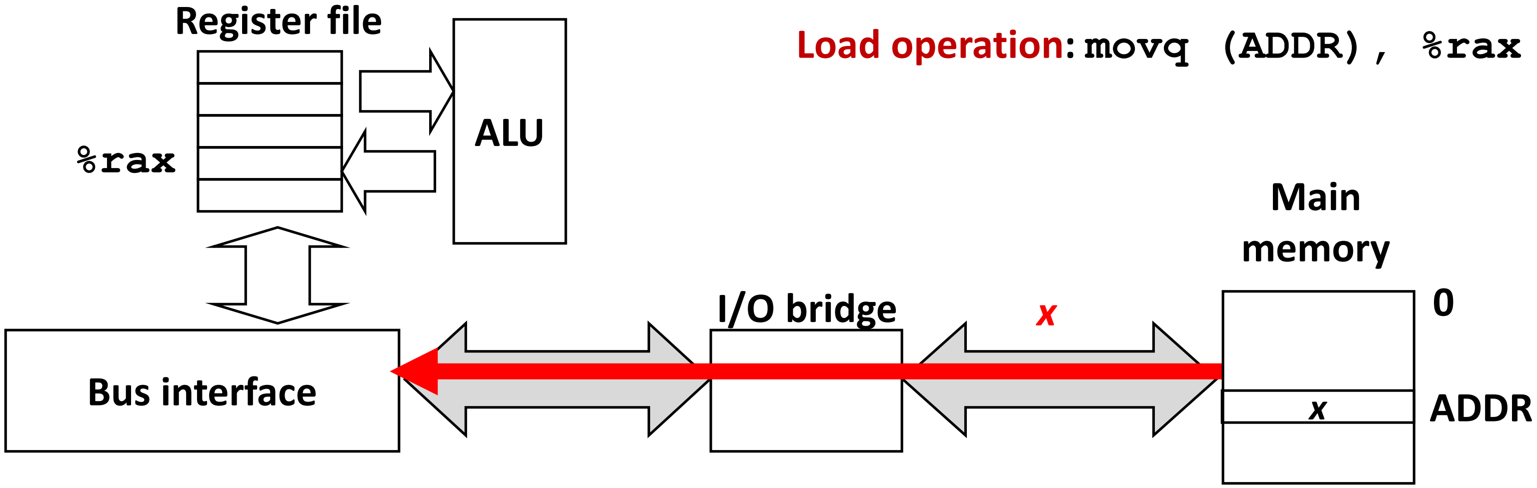
Memory Read Transaction (1)

- CPU places address ADDR on the memory bus.



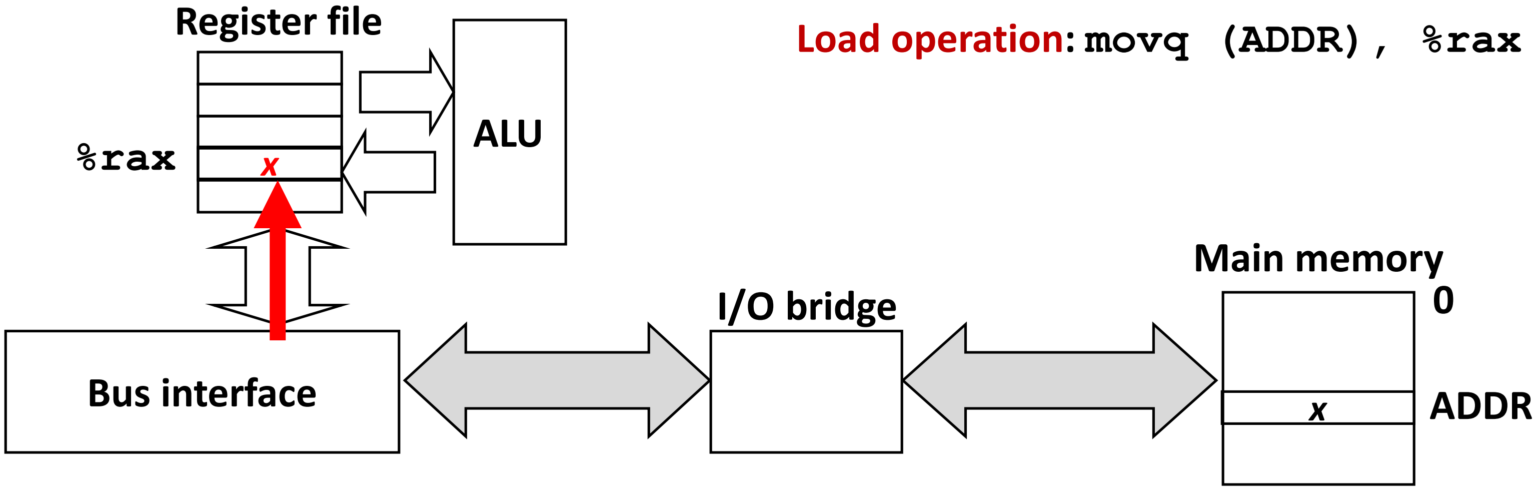
Memory Read Transaction (2)

- **Main memory reads address ADDR from the memory bus, retrieves word x, and places it on the bus.**



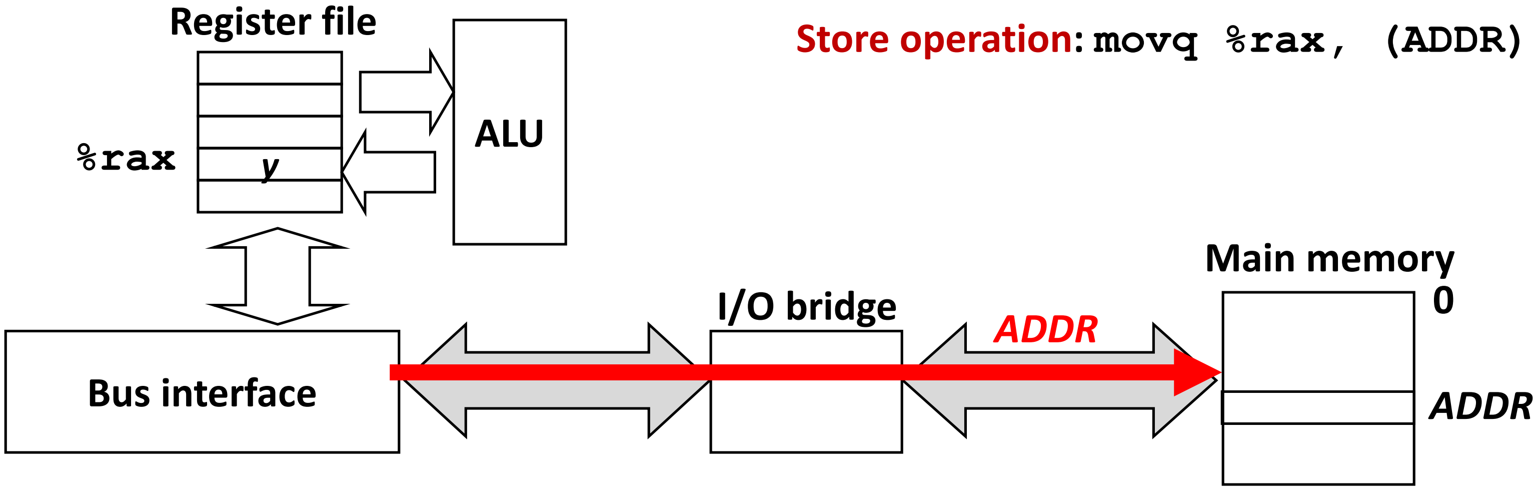
Memory Read Transaction (3)

- CPU read word x from the bus and copies it into register `%rax`.



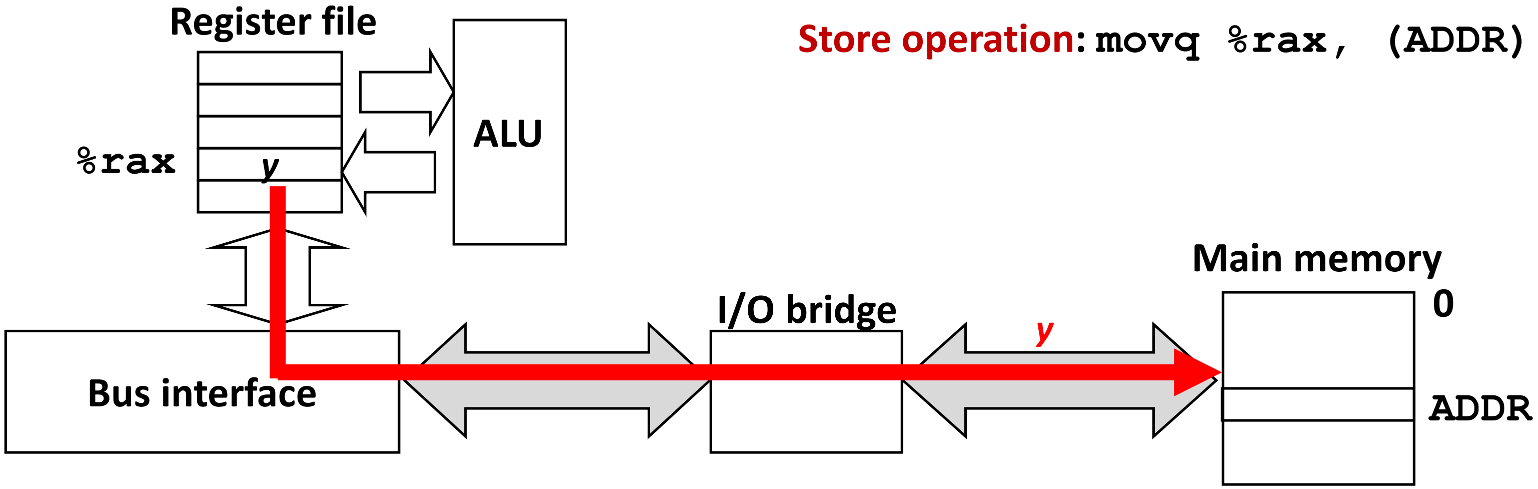
Memory Write Transaction (1)

- **CPU places address ADDR on bus. Main memory reads it and waits for the corresponding data word to arrive.**



Memory Write Transaction (2)

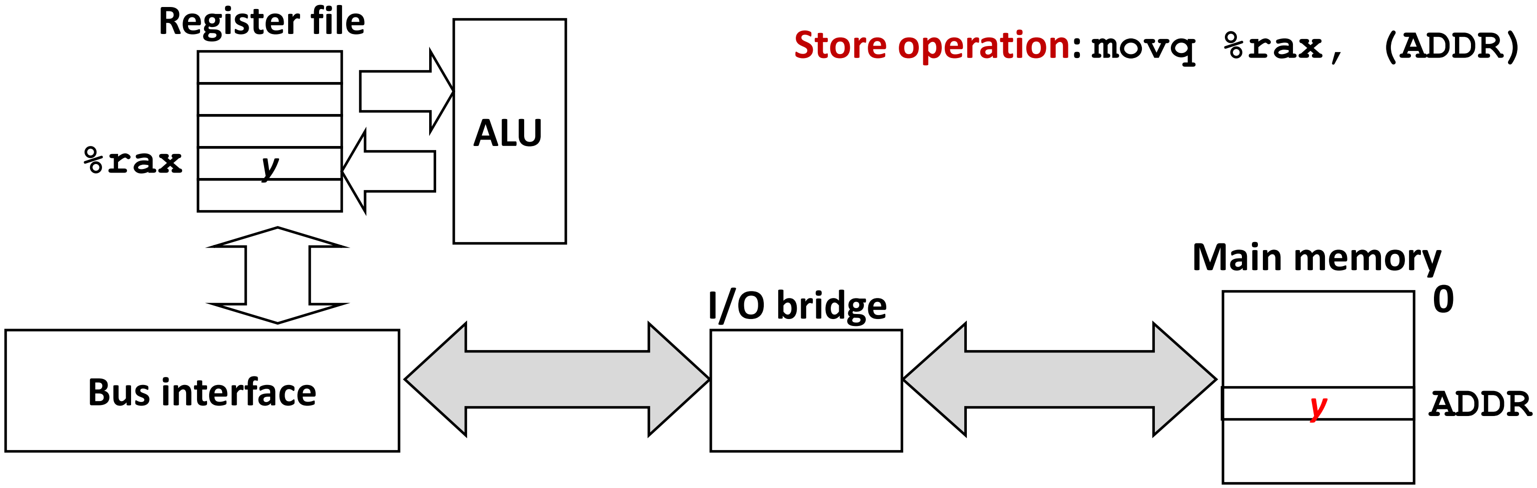
- CPU places data word y on the bus.



Store operation: `movq %rax, (ADDR)`

Memory Write Transaction (3)

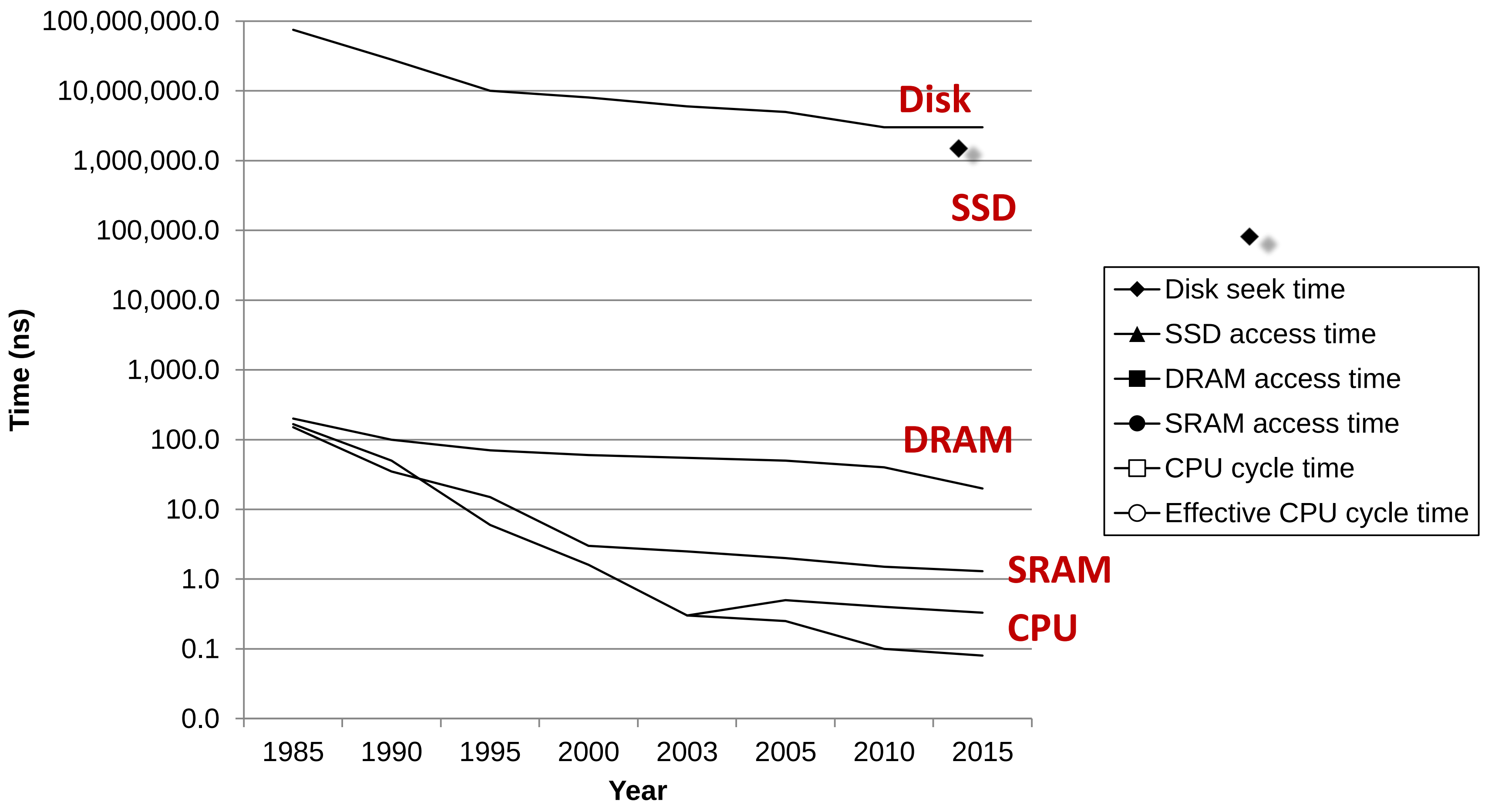
- Main memory reads data word y from the bus and stores it at address ADDR.



Today

- The memory Abstraction
- **Locality of reference**
- The memory hierarchy
- Storage technologies and trends

The CPU-Memory Gap

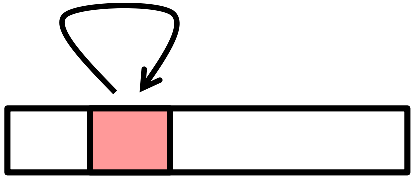


Locality to the Rescue!

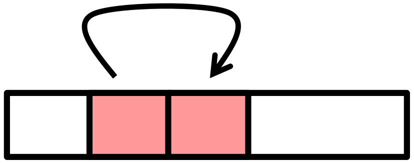
The key to bridging this CPU-Memory gap is a fundamental property of computer programs known as locality.

Locality

- **Principle of Locality: Programs tend to use data and instructions with addresses near or equal to those they have used recently**



- **Temporal locality:**
  - Recently referenced items are likely to be referenced again in the near future



- **Spatial locality:**
  - Items with nearby addresses tend to be referenced close together in time

Locality Example

```
sum = 0;
for (i = 0; i < n; i++)
    sum += a[i];
return sum;
```

	Spatial or Temporal Locality?
Data references	
Reference array elements in succession.	**spatial**
Reference variable `sum` each iteration.	**temporal**
Instruction references	
Reference instructions in sequence.	**spatial**
Cycle through loop repeatedly.	**temporal**

Locality Example (1)

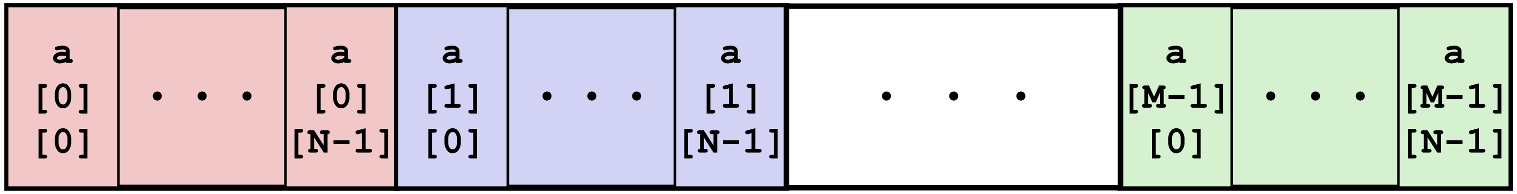
- **Question:** Does this function have good locality with respect to array a?

Hint: array layout is row-major order

Answer: yes

```
int sum_array_rows(int a[M][N])
{
    int i, j, sum = 0;

    for (i = 0; i < M; i++)
        for (j = 0; j < N; j++)
            sum += a[i][j];
    return sum;
}
```



Locality Example (2)

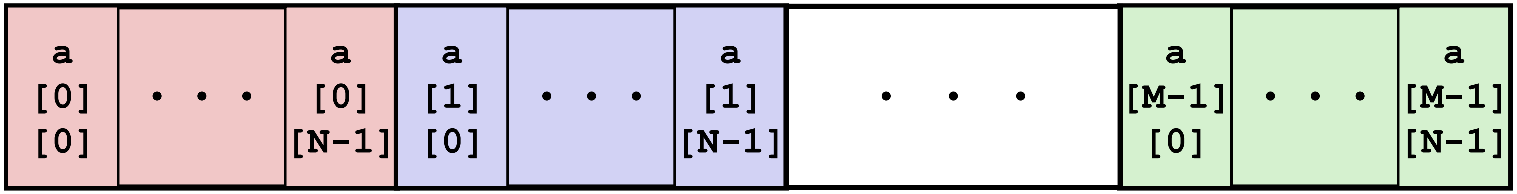
- **Question:** Does this function have good locality with respect to array a?

```
int sum_array_cols(int a[M][N])
{
    int i, j, sum = 0;

    for (j = 0; j < N; j++)
        for (i = 0; i < M; i++)
            sum += a[i][j];
    return sum;
}
```

Answer: no, unless…

M is very small



a [0] [0] | • • • | a [0] [N-1] | a [1] [0] | • • • | a [1] [N-1] | • • • | a [M-1] [0] | • • • | a [M-1] [N-1]

Today

- The memory abstraction
- Storage technologies and trends
- Locality of reference
- The memory hierarchy

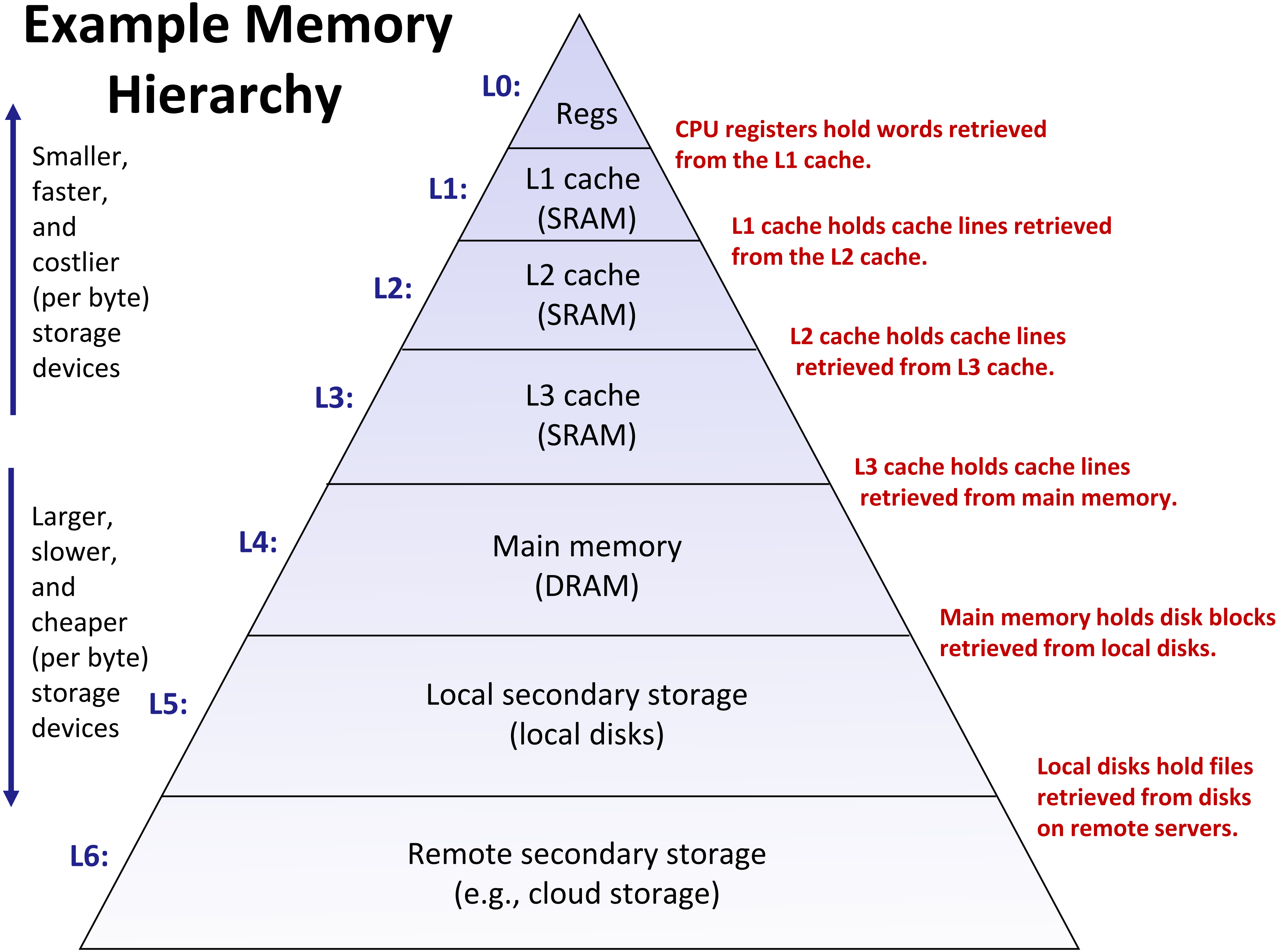
Memory Hierarchies

- **Fundamental properties of memory storage:**
  - The faster ➔ the more costs and the less capacity
  - The slower ➔ the less costs and the more capacity

- **These fundamental properties complement each other beautifully.**

- **They suggest an approach for organizing memory and storage systems known as a memory hierarchy.**

Example Memory Hierarchy


Smaller, faster, and costlier (per byte) storage devices
Larger, slower, and cheaper (per byte) storage devices
L0:
Regs
L1:
L1 cache (SRAM)
L2:
L2 cache (SRAM)
L3:
L3 cache (SRAM)
L4:
Main memory (DRAM)
L5:
Local secondary storage (local disks)
L6:
Remote secondary storage (e.g., cloud storage)
CPU registers hold words retrieved from the L1 cache.
L1 cache holds cache lines retrieved from the L2 cache.
L2 cache holds cache lines retrieved from L3 cache.
L3 cache holds cache lines retrieved from main memory.
Main memory holds disk blocks retrieved from local disks.
Local disks hold files retrieved from disks on remote servers.

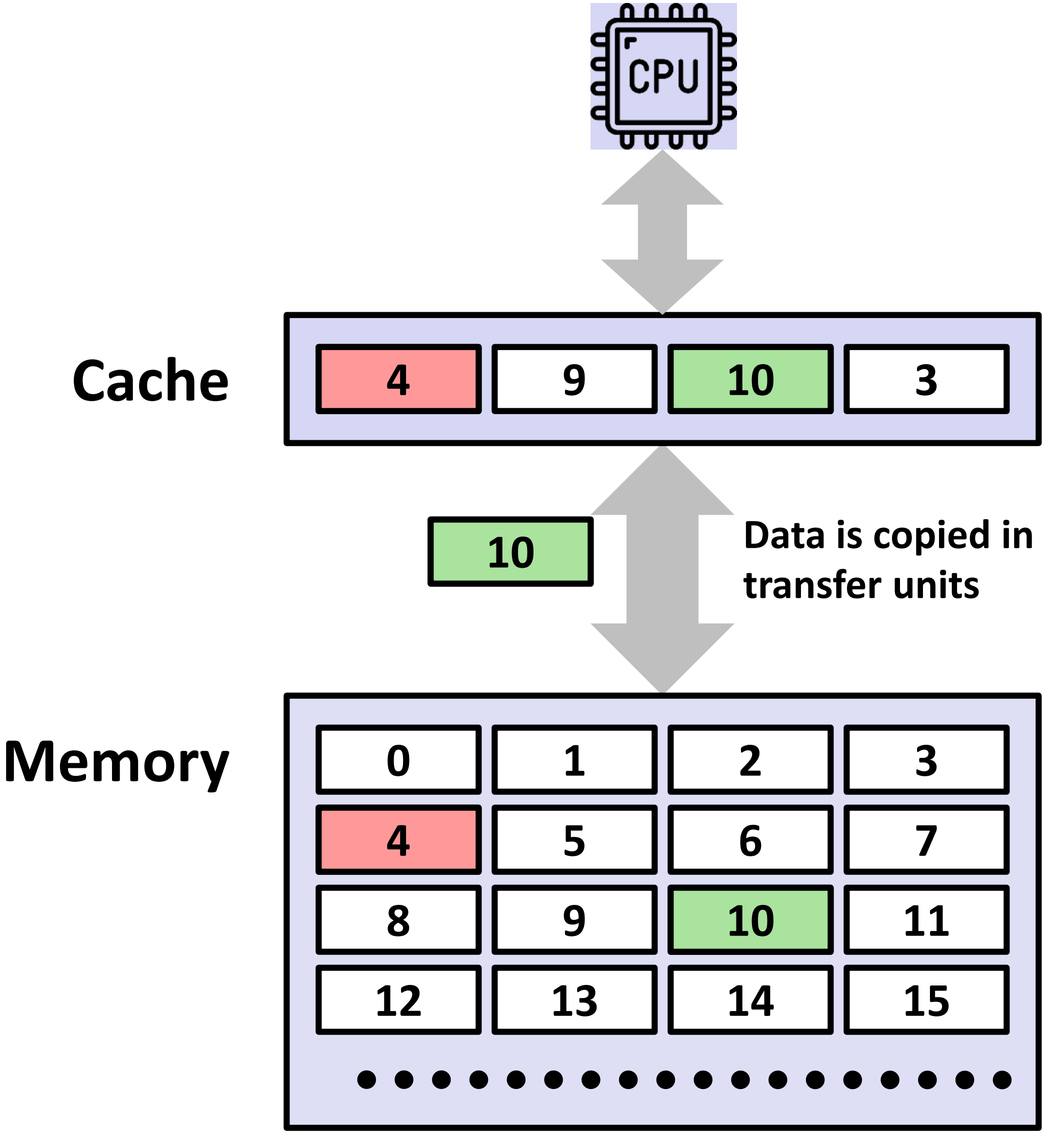
Caches

- ***Cache:* A smaller, faster storage device that acts as a staging area for a subset of the data in a larger, slower device.**

- **Fundamental idea of a memory hierarchy:**
  - For each k, the faster, smaller device at level k serves as a cache for the larger, slower device at level k+1.

- **Why do memory hierarchies work?**
  - Because of locality, programs tend to access the data at level k more often than they access the data at level k+1.

General Cache Concepts

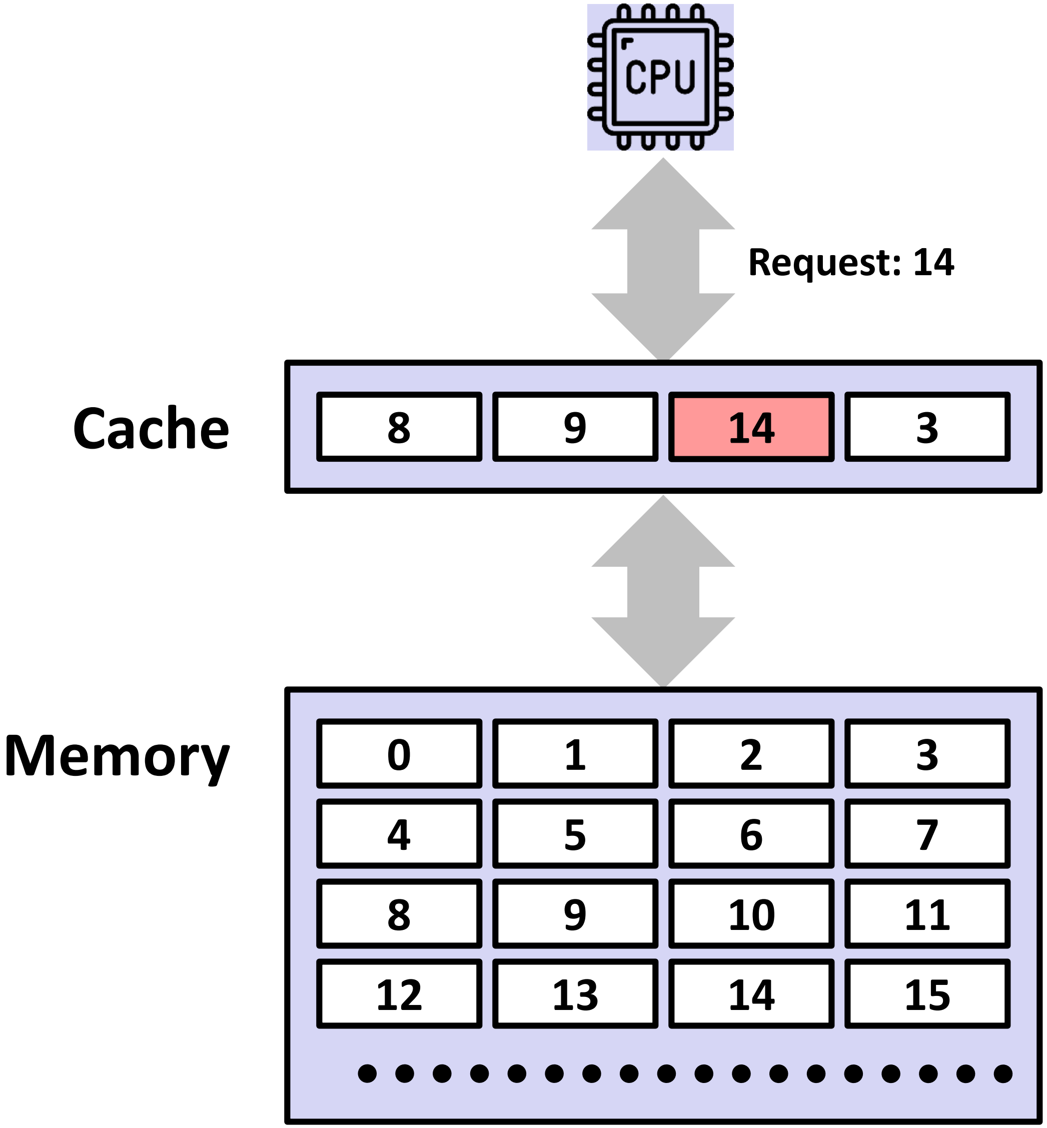


Smaller, faster, more expensive memory caches a subset of the blocks

Data is copied in block-sized transfer units

Larger, slower, cheaper memory viewed as partitioned into “blocks”

General Cache Concepts: Hit



CPU

Request: 14

Cache

8	9	14	3

Memory

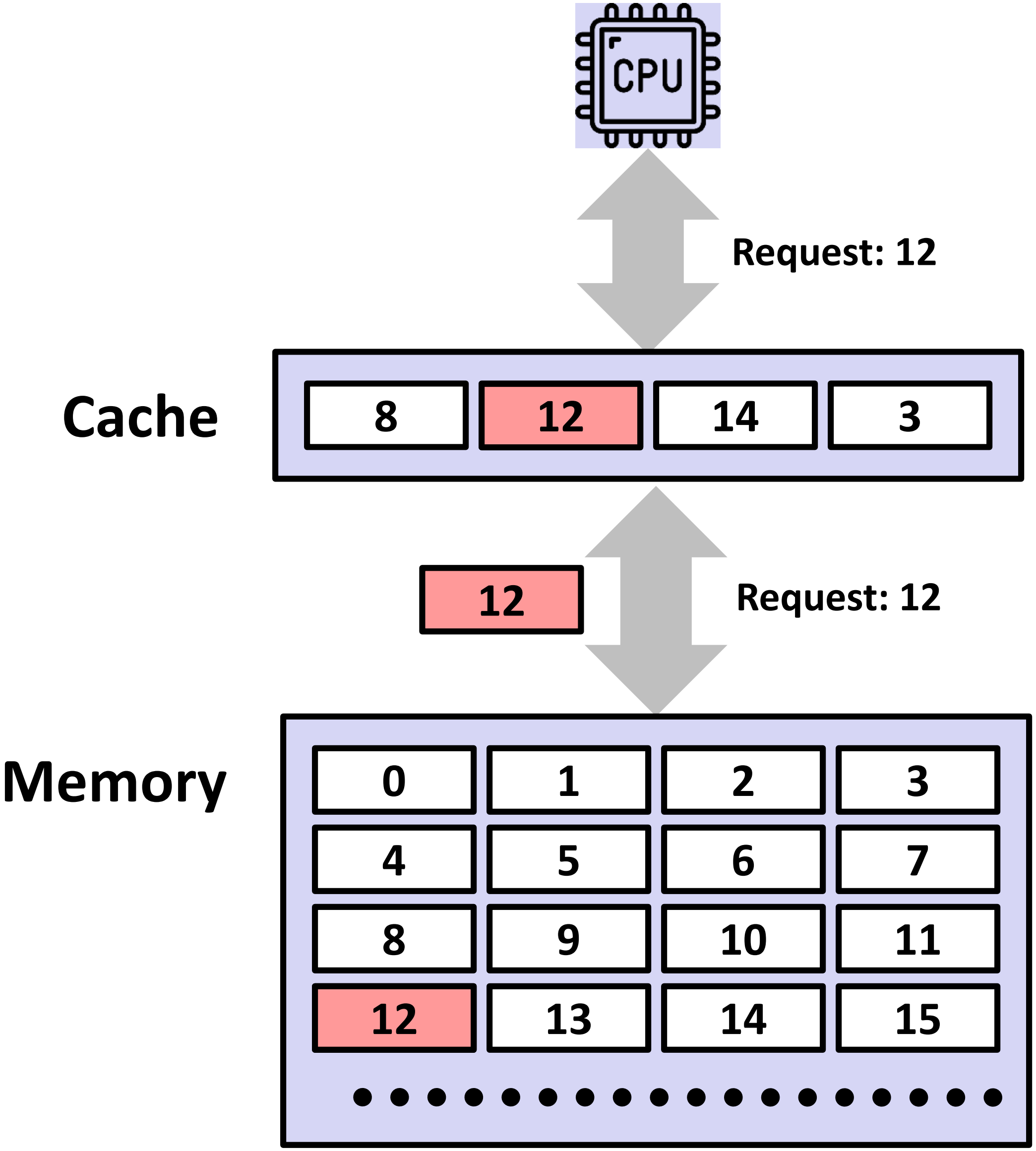
0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ●

Data in block 14 is needed

Block 14 is in cache:
Hit!

General Cache Concepts: Miss



Data in block 12 is needed

Block 12 is not in cache:
Miss!

Block 12 is fetched from memory

Block 12 is stored in cache

- Placement policy: determines where b goes (e.g., associativity)
- Replacement policy: determines which block gets evicted (victim) (e.g., FIFO/LRU)

Examples of Caching in the Mem. Hierarchy

Cache Type	What is Cached?	Where is it Cached?	Latency (cycles)	Managed By
Registers	4-8 byte words	CPU core	0	Compiler
TLB	Address translations	On-Chip TLB	0	Hardware MMU
L1 cache	64-byte blocks	On-Chip L1	4	Hardware
L2 cache	64-byte blocks	On-Chip L2	10	Hardware
Virtual Memory	4-KB pages	Main memory	100	Hardware + OS
Buffer cache	Parts of files	Main memory	100	OS
Disk cache	Disk sectors	Disk controller	100,000	Disk firmware
Network buffer cache	Parts of files	Local disk	10,000,000	NFS client
Browser cache	Web pages	Local disk	10,000,000	Web browser
Web cache	Web pages	Remote server disks	1,000,000,000	Web proxy server

Summary

- The speed gap between CPU, memory and mass storage continues to widen.

- Well-written programs exhibit a property called *locality*.

- Memory hierarchies based on *caching* close the gap by exploiting locality.